(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 May 2002 (16.05.2002)

PCT

(10) International Publication Number
WO 02/39486 A2

(51) International Patent Classification[7]: H01L

(21) International Application Number: PCT/US01/49984

(22) International Filing Date:
9 November 2001 (09.11.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/709,158 9 November 2000 (09.11.2000) US

(71) Applicant: NATIONAL CENTER FOR GENOME RESOURCES [US/US]; 2935 Rodeo Park Drive East, Santa Fe, NM 87505 (US).

(74) Agents: ROBERTS, Jon, L. et al.; Roberts, Abokhair, & Mardula, LLC, 11800 Sunrise Valley Drive, Suite 1000, Reston, VA 20191 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/39486 A2

(54) Title: INTEGRATED SYSTEM FOR BIOLOGICAL INFORMATION

(57) Abstract: A system for the integration of heterogeneous bioinformatics software tools and databases that allows interoperation of components adhering to a minimal set of standards. The system includes a software platform, one or more interface-based data models, and one or more component services. The invention utilizes an object oriented programming language to provide flexibility, synchronization, dynamic discover, and The Client Environment comprises a common user interface. Various embodiments disclose particular data models of use for bioinformatics and plant biology. The flexibility and improvements this invention provides over traditional object oriented approaches has use for other fields not concerned with bionformatics and biology.

INTEGRATED SYSTEM FOR BIOLOGICAL INFORMATION

BACKGROUND OF THE INVENTION

[0001] Biology is becoming increasingly dependent on computer software. Molecular and cellular biologists, geneticists, and biochemists rely on analysis programs, modeling tools, databases, and visualization software to accomplish their everyday work. As is true for many scientific fields, however, software development to support biology is more of a craft than a professional engineering discipline. Separately funded public and private groups independently create custom tools, often for narrowly conceived, short-term needs. Scientists engaging in computer programming, write software to support their research, but try to avoid being consumed by engineering concerns tangential to their main interests. In addition, the extremely rapid pace of scientific, technological and business changes in the field discourages the development of stable standards for interoperation and the formation of commercial companies that can deliver refined software at reasonable prices.

[0002] This disunity and decentralization that are natural and necessary in an active, developing scientific field unfortunately results in crippling incompatibilities among software tools and databases. Scientists find it slow, cumbersome, and labor-intensive to establish the connections across information resources that fuel scientific synthesis. At the same time, mounting pressure for "functional genomics," that is, identifying the functions of previously characterized genomic structures, makes the need for software and database compatibility even more acute. For example, comparative genomics is the technique of learning about one organism's genome through comparison to a better-studied relative is an important tool for functional inference. Comparative genomics often requires the integration of data from various specialized databases, and benefits from ready interoperation of those databases with assorted visualization and analysis tools.

[0003] **Figure 1** illustrates a typical sequence of steps a scientist might take to accomplish a relatively simple analysis task. To examine a DNA sequence in alignment with similar sequences from a public database, the scientist must use three different tools with three different interfaces and convert the output from each one to a format acceptable as input to the next. For more complex analysis, many other resources might be required. It has been

reported that more than half of a scientist's time may be spent on tasks related to the integration of data from incompatible databases and software programs. Moreover, this approach of converting the output of one program into acceptable input for another greatly limits the integrative potential between components that are highly interactive in nature.

[0004] Realizing that integration is essential, but that rapid change and limited development resources prohibit re-engineering of legacy components, researchers and commercial companies have tried various "bottom-up" approaches to allow interoperability of formerly incompatible resources. One common approach, used by both public systems and private application service providers (ASPs), is to provide a uniform Internet interface to various databases and analysis tools. These systems usually use common gateway interface (CGI) scripts or small Java programs that run on a server (Java servlets) to execute predetermined queries against databases, to make system calls to analysis programs, or to search file-based data repositories. Some systems allow users to send the output from one program into another program without concern for schematic representation or file formats. Many of these systems are ingenious and widely used, but all are limited by the internet-browser/CGI model. In addition, they generally permit little customization by the user and are not designed to allow "plug and play" of components.

[0005] A second approach, that emphasizes data access over analysis and visualization, is to enable complex declarative queries that span multiple heterogeneous databases. This approach has received considerable attention in bioinformatics, and has given rise to several systems. For example, U.S. Patent 5,859,972 to Subramaniam et al. discloses such a method. By use of a translation algorithm, a user query is simultaneously parsed to multiple databases. During the translation, the query is formatted for the destination database. However, such a method has inherent weaknesses as it either requires problematic on-the-fly mappings from representations in source databases to a definitive "ontology" (roughly, a global schema), or forces users to express queries in the sundry schemas of source databases. Moreover, it assumes a separation of user interface development and data integration that ignores the need for exploratory query formulation that users experience with inconsistent and unstandardized biological databases. Finally, these systems tend to be insufficiently flexible (e.g., to the addition or subtraction of

source databases or to changes in a global ontology) for such a turbulent field as the present state of bioinformatics.

[0006] A third approach is to package heterogeneous software tools and databases as components adhering to standard, well-defined interfaces, according to which information can be exchanged. This approach encourages components to encapsulate their differences and expose only minimal, abstract attributes and behaviors. Related computer programming models supporting this approach include Common Object Request Broker Architecture (CORBA) and Distributed Component Object Model (DCOM). CORBA is being promoted by Object Management Group (OMG), a consortium of over 700 companies. OMB publishes a set of standard interfaces using its Interface Definition Language (IDL) and implementation of components using CORBA. The component-based approach has a number of advantages for the problem of integration for biological information. For example, it enables interoperation with minimum homogenization and allows components implementing the same abstract interface to be interchanged. In bioinformatics, several groups have embraced this approach, including the CORBA based system for DNA sequence research disclosed in U.S. Patent 6,125,383 to Glynias et al.

[0007] The main stumbling block to this third approach is the rigidity imposed by standardization of interfaces. The OMG defines its standards by committee, in a way that is fair but also slow and arduous. It defines interfaces in relatively specific terms, which encourages exchange of data without loss of information, but makes standards harder to agree upon and more constraining for implementers. In addition, the specification of standard interfaces, useful as it is, does not address how components are to be integrated. System builders are free to integrate components as they see fit, and often write top-level controllers that instantiate and call components directly. In this way, they fail to take full advantage of the great potential for flexibility offered by component-based design, in contrast to the present invention.

[0008] As a result, what is needed in the field of bioinformatics is a system for the integration of heterogeneous bioinformatics software tools and databases that allows interoperation of components adhering to a minimal set of standards. As described below, the present invention satisfies these requirements and results in a highly flexible yet

powerful solution to advanced bioinformatics study that also takes advantage of existing heterogeneous components.

SUMMARY OF THE INVENTION

[0009] It is an object of the present invention to provide a system for the integration of heterogeneous bioinformatics software tools and databases that allows interoperation of components adhering to a minimal set of standards.

[0010] It is a further object of the present invention to use a common user environment.

[0011] It is another object of the present invention to allow client-side components to synchronize the behavior of their interfaces.

[0012] It is a still further object of the present invention to allow client-side components to use other client-side and server-side components to dynamically extend their functionality.

[0013] It is yet another object of the present invention to allow server-side components to use functionality of other server-side components.

[0014] It is an additional object of the present invention to allow components to be interchangeable, to eliminate explicit dependencies between interoperating components.

[0015] It is also an object of the present invention to allow maximum independence of bioinformatic components.

[0016] One embodiment of the present invention comprises a computerized system for integrating object based data models comprising a software platform, one or more interface-based data models, and one or more component services. The software platform comprises a Client Environment and a Client Bus. The software platform comprises software instructions in an object oriented programming language. The Client Environment comprises a common user interface. The Client Bus brokers all communication between software components and further comprises an Event Channel function and a Services Broker function.

[0017] The Event Channel enables components to register as listeners for particular types of events and react to them in prescribed ways.

[0018] The Services Broker allows components to request and provide services to one another by requesting a service by name, requesting services by types, or requesting services by attributes for which services are requested. Components, to the amount possible, interoperate without having a direct knowledge of one another. The data models comprise at least one data model useful for bioinformatics research.

[0019] Another embodiment of the present invention comprises a computerized system for integrating object based data models comprising a software platform, one or more interface-based data models, and one or more component services. The software platform comprises a Client Environment and a Client Bus. The software platform comprises software instructions in an object oriented programming language.

[0020] The Client Environment comprises a common user interface.

[0021] The Client Bus brokers all communication between software components and further comprises an Event Channel function and a Services Broker function. The Event Channel enables components to register as listeners for particular types of events and react to them in prescribed ways. The Services Broker allows components to request and provide services to one another by requesting a service by name, requesting services by types, or requesting services by attributes for which services are requested. Components, to the amount possible, interoperate without having a direct knowledge of one another. The data models comprise comprise at least one high-level map for an Integrated Bioinformation system.

[0022] A further embodiment of the present invention comprises a computerized system for integrating object based data models comprising a software platform, one or more interface-based data models, and one or more component services. The software platform comprises a Client Environment and a Client Bus. The software platform comprises software instructions in an object oriented programming language. The Client Environment comprises a common user interface. The Client Bus brokers all communication between software components and further comprises an Event Channel function and a Services Broker function. The Event Channel enables components to register as listeners for particular types of events and react to them in prescribed ways. The Services Broker allows components to request and provide services to one another by

requesting a service by name, requesting services by types, or requesting services by attributes for which services are requested. Components, to the amount possible, interoperate without having a direct knowledge of one another. The data models comprise at least one high-level map for an Integrated Plant Bioinformation system.

[0023] Additional objects and advantages of the present invention will be apparent in the following detailed description read in conjunction with the accompanying drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0024] **Figure 1** illustrates a schematic view of the typical steps a scientist might take for a simple bioinformatic research task.

[0025] **Figure 2** illustrates a schematic view of a simple embodiment of the system architecture of the invention.

[0026] **Figure 3** illustrates a schematic view of a simplified version of example interfaces of the invention.

[0027] **Figure 4** illustrates a screen view at a client user in accordance with one embodiment of the invention.

[0028] **Figure 5** illustrates a screen view at a client user in accordance with one embodiment of the invention wherein a pop-up window is displayed following dynamic discovery.

[0029] **Figure 6** illustrates a screen view at a client user in accordance with one embodiment of the invention wherein windows display synchronization aspects of the invention.

[0030] **Figure 7** schematically illustrates a high-level data map for a Map subsystem.

[0031] **Figure 8** schematically illustrates a high-level data map for a Sequence subsystem.

[0032] **Figure 9** schematically illustrates a high-level data map for a Metabolic Pathway subsystem.

[0033] **Figure 10** schematically illustrates a high-level data map for a Gene Expression subsystem.

[0034] **Figure 11** schematically illustrates a high-level data map for an Integrated Bioinformation system.

DETAILED DESCRIPTION OF THE INVENTION

[0035] As a preliminary matter, the following breif *glossary of terms and acronyms* is provided to aid in understanding of the disclosure.

BLAST -- Basic Local Alignment Search Tool is a set of similarity search programs designed to explore all of the available sequence databases maintained by the National Center for Biotechnology Information (NCBI), regardless of whether the query is protein or DNA.

CGI - Common Gateway Interface - Set of rules that describe how an Internet server communicates with another application running on the same computer and how the application (called a CGI program) communicates with the Internet server. Any application can be a CGI program if it handles input and output according to the CGI standard.

CORBA-- Common Object Request Broker Architecture--an architecture neutral, object oriented client-server solution. With CORBA you can abstract an object by its services and publish these using the IDL (Interface Definition Language). A client can then connect to and use these services.

Component - a reusable program building block that can be combined with other components in the same or other computers in a distributed network to form an application. Examples of a component include: a single button in a graphical user interface, a small interest calculator, an interface to a database manager, or a statistical analysis tool.

Compounds - their elemental and structural formulas, molecular weights and other physical and chemical properties.

Gene - unit of inheritance, a piece of the genetic material that determines the inheritance of a particular characteristic, or group of characteristics. Genes are carried by chromosomes in the cell nucleus and are arranged in a line along each chromosome.

Genome - the complete collection of an organism's genetic material. The human genome is composed of an estimated 50,000 to 150,000 genes located on the 23 pairs of chromosomes in a human cell.

Homology - the relationship among sequences due to descent from a common ancestral sequence.

Instantiate - to create a particular realization of an abstraction, for example, defining a particular variation of object within a class, giving it a name, and locating it in some physical place.

Java - an object oriented programming language designed for use in the Internet. Java can be used to create applications that may run on a single computer or be distributed among servers and clients in a network.

Java servlet - a small program that runs on a server, but invoked by a client request. The server must be running Java.

Metabolic steps - chemical reactions and transport steps, including substrates, products and their stoichiometric coefficients.

Metabolic proteins - enzymes and transport proteins, including physico-chemical properties of these peptides, kinetic modifiers and external links to sequence and structural databases.

Pathways - the sequences of steps that compose each pathway, including their direction, catalyst (if any - uncatalyzed steps are also supported), modifiers, and location information (at organism or intracellular level).

QTL – quantitative trait locus.

Taxonomy (or, more properly, Systematics) - the classifying of organisms into a series of hierarchical groups. The criteria that different taxonomic systems use for defining these groups may be very different (morphological, molecular, etc.), but in general these groups tend to be reflective of common evolutionary descent. Taxonomic studies may seek to understand the known similarities and differences between species based on their evolutionary history, or they may

attempt to use known attributes from a highly studied species to postulate unknown characteristics.

[0036] The present invention comprises an integrated, object-based, computing software platform, hereinafter referred to as the "integrated system" or "IS," that enables genomic researchers and bioinformaticians to access and utilize disparate bioinformatic software tools and data sources in a seamless, unified environment. The disparate software tools and data comprise "components" that are integrated by the invention. For purposes of describing this invention, a "component" is a coherent package of software that can be independently developed and delivered as a unit, and that defines interfaces by which it can be composed with other components to provide and use services. For example, a gene expression component may include a relational database engine, a suite of analyzing tools, and an upload engine for storing, analyzing and downloading gene expression data.

[0037] The present invention will be discussed using the object oriented system architecture standards supported by OMG and commonly known as Common Object Request Broker Architecture (CORBA). This is not to be construed as a limitation on the present invention as numerous object oriented computer environments exist and continue to evolve. For example, it will be readily apparent to those skilled in the art that DCOM (Distributed Component Object Model) or other object-oriented methodology can be equally employed. For simplification of this disclosure, CORBA terminology will be utilized to provide a common meaning of terms.

[0038] Although not required for a full understanding of the present invention, the "ISYS Development Manual" (which has descriptions and programming details for programming and implementing IS components and services) is incorporated by reference herein and is available from the National Center for Genome Resources, Santa Fe, NM.

[0039] The present invention includes a novel software platform integrating bioinformation data models. One embodiment of the present invention's software platform is partially illustrated in **figure 2**. In particular, **figure 2** schematically illustrates a portion of the system architecture. It shows several components on the client side, integrated by way of the "IS Platform". Additional aspects of the invention include the platform, the

exchange of services, the exchange of events, how non-platform components fit into the system, and the data model that unifies the system.

[0040] The "IS Platform" comprises two components that are responsible for the integration and management of all others: the *ClientBus* and the *IS Client Environment (ICE)*. The ICE is the simple user interface that allows a user to invoke components directly. The ClientBus is responsible for all communication between components. It further comprises a combination of an Event Channel (sometimes called a bus) and a Broker. As an Event Channel, the ClientBus enables components to register as "listeners" for particular types of events, or occurrences, and react to them in prescribed ways. As a Broker, it allows components to request and provide services to one another, as defined by service name, or by types and attributes of objects for which services are requested. In both cases, components interoperate without having direct knowledge of one another, to promote flexibility.

[0041] The interface to the ClientBus provides: (1) registry of a component as a listener for a particular class of events; (2) broadcast of an event, generally linked to a specific underlying data object; (3) registry of a component as a provider of particular services; (4) request of services by name; and (5) request of services applicable for a particular object. Of these, the first two relate to events, and the final three to services.

[0042] Event exchange is a mechanism to allow synchronized behavior among components. The Event Channel of the invention is a variant of the standard Observer pattern that allows components to exchange events without registering directly with one another. Instead they remain decoupled and interact only with the Event Channel, which behaves as a mediator among components.

[0043] In the present invention any component may register with the bus as a listener for any type of event. Event objects generally are given references to associated data objects. Once a component is registered, all events of the specified type will result in calls to the component's listener method, as long as the two components are defined as being synchronized. The invention additionally allows deactivation of synchronization where it is unnecessary or confusing.

[0044] For example, in **figure 2**, when a user selects a matching sequence in the SimilaritySearcher, that component generates an ItemSelectionEvent and "fires" it by calling a method on the ClientBus. This event holds a reference to a data object representing the selected homolog. If the SequenceViewer is synchronized with the SimilaritySearcher (for example, if synchronization requested by the SequenceViewer Factory when it originally spawned the SimilaritySearcher) then the ClientBus will call the SequenceViewer's registered listener for ItemSelectionEvents. Finally, this listener, within the SequenceViewer, inspects the data object for an identifier, locates the corresponding object among the features it is displaying, and highlights it as selected.

[0045] The other aspect of the ClientBus is the Broker. As embodied in the invention, the Broker of the ClientBus allows decoupled components to interact. It comprises a "pull" mechanism in that the receiving component is provided services when it requests them. Traditionally, a Broker follows the CORBA standards and generally is applied to systems having distributed components.

[0046] The Broker of the present invention differs from a traditional CORBA broker in three main ways. First, services are objects, passed via the broker from service providers to service consumers. They encapsulate their behavior using the Command pattern. This is a relatively minor change that allows services to be passed among components, and that causes specific services (analogous to methods), rather than servers with specific interfaces (collections of related methods), to be the unit of exchange.

[0047] Second, the Broker runs on the client, and allows direct interoperation only among client-side components. As a result, client-side and server-side proxies handling tasks associated with network communication, are not necessary.

[0048] Third, in addition to exchange of services by name, the ClientBus supports Dynamic Discovery, by which registered service providers evaluate a particular data object and respond whether or not their services are suitable for it, and suitable services are forwarded to a requester.

[0049] To understand the mechanics of Dynamic Discovery, consider how a user can interact with the embodiment of **figure 2**. Assuming the SequenceViewer Client has identified a data object associated with the selected graphical object. When a user invokes

a request for all services capable of operating on this object, a service request is transmitted to the ClientBus. The ClientBus has forwarded the request to all registered service providers. As shown in **figure 2**, the service providers include the SequenceServer Proxy and the SequenceAnalysis Proxy. These service providers inspect the object, and if it finds its type recognizable and sufficient data present, creates appropriate Service objects and passes them to the ClientBus. The ClientBus returns these service objects to the SequenceViewer Client. In a typical embodiment, the SequenceViewer Client displays their descriptions in a pop-up menu. The user can then select one of these descriptions, which when selected, invokes the corresponding service, passing it the original data object.

[0050] Non-platform components directly belonging to the system (i.e., components that are neither part of the platform nor external servers) must implement at least one of two interfaces: *Client* and *ServiceProvider*. Implementers of Client may fire and receive events, and implementers of *ServiceProvider* may register and provide services. Any component may request a service. Also, a single component may implement both interfaces (although this is not usually done).

[0051] ServiceProviders usually act as server proxies or client factories. Server proxies are gateways to servers. They provide a layer of insulation between the system and external resources. This is valuable in two ways: if the interface to an external resources changes, alterations to the system may be confined to the server proxy; and server proxies that fulfill the same abstract responsibilities (e.g., a server proxy that executes BLAST searches at the National Center for Genome Research, and one that executes BLAST searches at National Center for Biotechnology Institute) can be interchanged without perturbing the rest of the system. Client factories (service providers) are responsible for generating instances of clients.

[0052] Servers are necessary for the system but are external to it. Any resource that can be called by a server proxy, generally as a provider of data, can function as a server. Such a resource may exist on the Internet, on a local network, or on the same machine as the client. (In general, the problem of scalability can be addressed this way: processes involving large quantities of data or intensive communication are handled on appropriate

servers, which are integrated in the client-side environment via server proxies). Communication with remote servers can occur using any common protocol including HTTP, CORBA, and Java RMI™.

[0053] Separately-developed software tools generally are added to the system as clients or servers. In one embodiment of the invention user interface components are added by wrapping them with a simple, lightweight Java class that implements the appropriate interfaces and delegates calls to the legacy tool. If the tool is to invoke services or fire events, its user interface events must be intercepted in some way. The methods to accomplish this are well known to those familiar in the art, particularly if the component is written in Java and its source code is available, or if a sufficient Applications Programming Interface (API) is provided. If not, intercepting events is still possible although more difficult.

[0054] An accompanying client factory (service provider), also small and simple, must be created to support the client. Data providers, either in the form of databases or analysis tools, are generally added as servers, and accessed via server proxies. Since server proxies have complete freedom in the retrieval of information, they may fulfill their responsibilities without using a true server (in the conventional sense of an interacting local or remote process).

[0055] For example, to retrieve information a server proxy may invoke methods of a utility library, or spawn a child process.

[0056] As has been discussed above, the software platform of the invention adds powerful features of flexibility and Dynamic Discovery to traditional approaches for integrating heterogeneous data components and services. Preferred embodiments of the invention apply the integrated software platform to data models of particular usefulness for bioinformatic research.

[0057] Preferably the software platform is applied to data models that take advantage of the fact that there exist a relatively small number of fundamentally important classes of biological objects that tend to resist major schematic variation: e.g., DNA sequences, genes, proteins, enzymes, pathways, maps, mappable elements, taxonomic designations. Along with their core attributes, these classes form a common denominator for biological

databases, with differences among schemas tending to occur in less essential details (e.g., how feature locations are represented, what kinetic parameters are represented for enzymes).

[0058] Because these fundamental classes tend to reflect stable and established models and concepts, relationships between components generally depend on them, not on the lesser details (their identities become evident when one analyzes the needs of each component from the others). For example, the entities in GSDB (Genome Sequence Database maintained by National Center for Genome Resources) and PathDB (Metabolic Pathways Database maintained by the National Center for Genome Resources) are associated primarily according to relationships between genes and enzymes, and the data entities in GSDB and TAIR (The Arabidopsis Information Resource maintained by the National Center for Genome Resources and Carnegie Institution of Washington) are primarily associated according to relationships between maps and sequences.

[0059] The preferred embodiment of the invention exposes only the fundamental classes in component interfaces, and encapsulates manipulation of the less important details within the components. The cost of this is small, compared to the advantages it offers in system flexibility, because the details tend not be important across component boundaries.

[0060] For example, a search for enzymes having particular kinetic properties can be encapsulated within a pathway-searching component. When looking across components to find genes in GSDB associated with the particular enzymes, only the relationship between the fundamental classes for enzymes and genes is used. The kinetic parameters are generally important only indirectly for cross-database uses, so they are not exposed in the pathway component's interface, to avoid unnecessary interdependencies with other components. Java interfaces have been found to support implementation of this approach.

[0061] An interface can be defined for each of the fundamental classes, representing its attributes (via accessor methods) and essential behaviors. Interfaces can inherit multiply from one another, classes can implement multiple interfaces, and interfaces can include methods that declare associations with other interfaces. Consequently, a single object can simultaneously represent multiple fundamental classes.

[0062] **Figure 3** illustrates an example prepared using Java. A class called *Homolog*, used by the SimilaritySearcher, implements both the fundamental interface *IsysSequence* and an interface *HasTaxon* that declares a close association with an object having the fundamental interface *IsysTaxon*. When the user performs Dynamic Discovery on an object of class Homolog services will respond in terms of both interfaces so that the object can be manipulated either as a sequence or as a taxon. Similarly, when an event is fired that refers to such an object, other components will react to both the corresponding IsysSequences and IsysTaxons.

[0063] This novel approach differs from the standard Model-View-Controller (MVC) approach described in *A System of Patterns: Pattern-Oriented Software Architecture*, written by Buschmann, et al. Instead of observing a shared model, components exchange information about private models and views in their Java interfaces (components can still use the MVC pattern internally). Since they can implement these interfaces however they choose, private models can be kept at a high-level or made arbitrarily rich. For example, the TaxonomyBrowser of **figure 3** has a much richer implementation of the interface IsysTaxon than does the SimilaritySearcher; but because certain objects in both components are understood to be of type IsysTaxon, they can be exchanged readily. Furthermore, the TaxonomyBrowser can change its implementation however it wants without disrupting the SimilaritySearcher, as long as the interface is kept the same.

[0064] The resulting data models of the invention do not represent a competing standard for representing bioinformatics data, but are compatible with developing standards (Life Sciences Research Task Force, 1997). The reason is that the data models of the invention are able to be more abstract and less comprehensive than other models and can be easily mapped to subsets of them. Components may employ more detailed data maps for intra-component use, while simultaneously communicating with other components via higher-level data models. They may even choose to interact with one another directly at lower levels at the cost of decreased insulation from changes to one another.

[0065] Following early work at implementing example data models, an improvement was incorporated that increases the flexibility of the invention. Coding of fundamental classes into the interfaces prevented different components from using different "views" of core

attributes. As a result, the preferred embodiment of the invention associates attributes of data models with fundamental classes at run-time by aggregation. Consequently, the set of attributes belonging to objects of a given type will not be fixed, and components will not need to be recompiled when a data model is extended.

[0066] **Figures 4, 5,** and **6** illustrate the user interface displays that result when a user of the present invention is performing the same analysis steps executed in **figure 1. Figures 4, 5,** and **6** depict two components, *SequenceViewer* **100** and *SimilaritySearcher* **104**. The SequenceViewer **100** displays colored bars **102** representing a DNA sequence and its annotations (parcels of information about sequence segments based on laboratory experiments or computer analysis). The sequence is represented at top (off the screen here), with the 5' end at left and the 3' end at right. The annotations are tiled beneath it, each bar color representing a different type (e.g., gene, exon, intron, transcription binding site, and region of high similarity to another sequence). The tool lets users scroll and zoom the display and view detailed description of annotations. In **figures 4, 5,** and **6** all visible bars represent regions of high similarity to other sequences (i.e., all are of the same type; the ones of darker color are simply selected).

[0067] The SimilaritySearcher **104** launches searches of single query sequences against large databases of "background sequences" and displays search results. **Figures 4, 5,** and **6** show the SimilaritySearcher's results browser (i.e., the search has already been launched and the results returned). The top pane **106** of the browser is a summary table of the search results, each row of which represents a single matching sequence from the background database. The bottom pane **108** displays details about the first selected match, including an alignment of the query sequence and background sequence in the regions of high similarity. The search was executed using a popular program called *BLAST*.

[0068] The components SimilaritySearcher and SequenceViewer are separate and have no direct dependencies on one another; nevertheless, they appear to the user to be closely integrated. In this embodiment of the invention, the SimilaritySearcher **104** was invoked from the SequenceViewer **100**, which passed it the text of the displayed sequence for use as a query sequence (using the present invention, the user began in the SequenceViewer **100**, rather than with a text file as in **figure 1**. The annotations **112**, visible in **figure 5** in

SequenceViewer **100** were not originally present; they reflect the similarity search hits in the other component, and only appeared when the search returned and the browser appeared. Selection and visibility of the search hits are synchronized in the two components. When the user selects items in the table in the top pane of the SimilaritySearcher **104**, the corresponding annotations **112** are selected in the SequenceViewer **100** (as seen in **figure 5**). Similarly, when the user causes items to disappear in the SimilaritySearcher **104**, they disappear also from the SequenceViewer **100**. This can be seen in **figure 4**, which shows the application of a filtering tool **110** to the similarity search results (the tool filters by BLAST's "expect value", a statistical score indicating the significance of sequence similarity), and the corresponding disappearance of annotations in the SequenceViewer **100**. All of this synchronization occurs via the broadcast and reception of generic events. There is no analog to this synchronous behavior in the execution sequence of **figure 1**.

[0069] The pop-up menu **114** obscuring part of the SimilaritySearcher **104** in **figure 5** appears when the user clicks the right button of the mouse after selecting several of the rows in the search-results table **106**. The options appearing in this menu **114** are dynamically generated and can be restricted to the set of components that a user installs and registered with the integrated system security controller. Using a process denoted as *Interactive Discovery*, registered components are interrogated about the selected data set. Those that respond that they can operate on the data set are represented in menu **114** with appropriate descriptions.

[0070] Interactive Discovery is an important aspect to the present invention. This mechanism encourages an exploratory mode of usage in which new paths through the bioinformatics system emerge dynamically, according to selected data and what components are present. If the user selects the option to "Perform multiple sequence alignment" from menu **114**, he is presented first with interface window **116** of **figure 6** to a component program that computes a multiple alignment of the selected sequences. This, in turn invokes interface window **118** to a component program that provides an editable display of the alignment.

[0071] This example illustrates how various components in this embodiment of the invention interact in an integrated fashion. First, the program invoked through window **116** of **figure 6**, which is available as a computer program written in C, is wrapped with a service provider, a component that registers abstract services with the integrated system but hides how they are implemented. Second, the program invokes through window **118** of **figure 6**, available as a Java application, is wrapped with a simple Java class that makes it appear to the integrated system like any other client. This Java wrapper handles the broadcast and reception of integrated system events and transmits them in terms the program can understand. Third, the full-length sequences required for the multiple sequence alignment, not available in SimilaritySearcher **104** (it knows only of segments of background sequences), are retrieved from a public database of DNA sequences. This occurs transparently to the user by way of another application service provider that acts as a proxy to a resource available on the Internet.

[0072] The advantages of this aspect of the invention as compared to the scenario in **figure 1** extend beyond convenience and efficiency. Although the user in this example follows an anticipated course of action with the integrated system, Interactive Discovery suggests paths that might otherwise be omitted. In addition, the integrated system provides new capabilities and new perspectives on biological data. For example, when a user filters the search results (item **110** of **figure 4**) and observes the effect in the SimilaritySearcher **100**, the user is able to view the physical locations of search results above a certain threshold—something permitted by neither component individually.

EXAMPLES

[0073] **Figure 11** illustrates one embodiment of an integrated data model useful with the invention. **Figure 11** is a composite of four separate high-level data models represented in **figures 7, 8, 9,** and **10**. Fundamental linkages and relationships are highlighted in **figure 11** with thicker lined representations. For example, Metabolite shows as a link between the data model for Metabolic Pathways of **figure 9** and the data model for Gene Expression illustrated in **figure 10**.

[0074] **Figure 7** illustrates a data model for a Map Subsystem. This subsystem deals with high-level linear representations of genetic information, e.g. chromosome-level

descriptions of the placement of genetic markers. Its basic entities are Maps **40** and MappableObjects **42**. A Map represents a linear segment of a biological macro-molecule such as DNA or protein which may or may not be characterized in terms of the exact sequence of its subunits, but upon which entities that are "mappable" (by some method such as recombinant crossing experiment) have been assigned location. A MappableObject **42** is an abstraction representing anything that can be located on a Map, e.g. genetic markers, QTLs, genes. A MappableObject **42** is also the linkage point between the Map subsystem, and other subsystems, whose elements may be "mappable."

[0075] The same MappableObject can appear on many different Maps, and a Map is comprised of many different MappableObjects. This relationship is affected through the MappedObject **44** class, which assigns location to a MappableObject in the context of a given Map.

[0076] The Map itself inherits from Mappable Object, meaning that Maps may be recursively constructed out of other maps. This is one of the most powerful features of the Map subsystem, as it means that MappedObjects on different Maps may be related to one another by having their Maps located with respect to one another on another Map. For example, a Chromosome can be a Map comprised of smaller Maps of regions that have been individually characterized. The objects in the individual regions can be related to the Chromosome by means of their location on their own Map, and its location on the Chromosome map.

[0077] There are two major subclasses of Map. A genetic map **46** represents distances in terms of recombination probabilities, while a physical map **48** represents distances in terms of base pairs. In the Sequence subsystem of **figure 8**, all sequences inherit from PhysicalMap to allow their features to be specified as MappedObject with base pair coordinates.

[0078] **Figure 8** illustrates the Sequence subsystem **50** data model. This subsystem deals with genetic and protein information in terms of its encoding as a linear string of characters. Its major entities are the sequences themselves (DNA **56**, RNA **54**, and AA **58**) and characterizations of functional features of sub-spans of these sequences, such as genes, motifs, repeats, promoters and binding sites. Another major responsibility of the

Sequence subsystem **50** is the representation of homology relationships **52** between sequences. In general, these are based on alignments of sets of sequences, however other subsystems will generally only be interested in the existence and level of homology between entities that have sequence-based representations.

[0079] The relationship between Sequences and their features is represented in terms of the Map subsystem's data model; Sequences are considered as a kind of PhysicalMap, and their Features are MappedObjects that are located with respect to the Sequences as MappedObjects. The use of a common data model stems from the fact that both subsystems are representing linear entities.

[0080] Some of the details that are managed by the Sequence subsystem, but hidden from view of other subsystems include the representation of confidence in the characters of the sequence, as well as other details relating to sequencing experiments such as the libraries used to obtain the clone. Another major detail that is best hidden from other subsystems is the fact that entities such as genes are often represented by multiple sequences, and are typically either only partially represented by or represented as a subspan of a given Sequence. Preferably these facets of the sequencing process are hidden from other subsystems, so that if another subsystem asks for the "sequence" of a given gene, it may receive a consensus sequence representing a composite view of the subspans of all sequences that include this gene.

[0081] **Figure 9** illustrates the high level data model for the Metabolic Pathways **60** subsystem. This subsystem deals with metabolism, i.e., the networks of transformations of chemical compounds that are catalyzed by genetically encoded enzymes. Its basic entities are Metabolites **62**, MetabolicSteps **64**, Catalysts **66**, and MetabolicPathways **60**. Metabolites are chemical compounds that are transformed into other compounds by MetabolicSteps. MetabolicSteps may be spontaneous, or they may require Catalysts to proceed under the conditions found in living cells. These Catalysts are proteins **68** that are encoded by genes. Finally, MetabolicPathways **60** are connected sets of MetabolicSteps **64**, in which the products of one step become the substrates of another step, and so on. These networks of chemical transformations can be quite complex, and can include MetabolicSteps **64**, which transport Metabolites **62** to different locations in the organism.

[0082] In general the Metabolic Pathway subsystem represents metabolic potentials that exist in organisms; that is, its MetabolicSteps **64** represent chemical transformations that can take place under the appropriate conditions (e.g., if substrates are present and enzymes are expressed), and MetabolicPathways **60** represent networks of reactions that can exist, provided that all the reactions do take place under the same conditions. This information is of interest in itself and has application to fields such as Metabolic Engineering. However, it is also useful to assess which reactions and which pathways actually do exist in specific organisms under specific conditions. Although the Pathway subsystem does make some attempt to represent this information, complete information regarding the state of living systems currently is conceived as belonging to the Expression subsystem.

[0083] The Metabolic Pathways subsystem contains much internal detail regarding the kinetic and thermodynamic properties of reactions which are useful for modeling tools; however, this information should not be exposed to other subsystems, as it is relatively complex and of little direct usefulness elsewhere.

[0084] Metabolic Pathways **60** links to other subsystems most strongly through Catalysts **66** that are proteins **68** (encoded by genes) represented in terms of their catalytic function. Both Catalysts **66** and Metabolites **62** will also have connections to the Expression subsystem.

[0085] **Figure 10** illustrates the data model for the Expression subsystem. Generally, the Expression subsystem tries to represent the state of cellular systems under specific conditions. In this subsystem, state is represented in terms of levels of abundance of mRNA transcripts (GeneExpression), Proteins (Proteonics), and Metabolites (Metabolomics). These levels will be assayed via Experiments that will generate Profiles of the level of expression of these substances under various Conditions. For example, a Gene Expression experiment **72** may generate Profiles for a set of genes **80** in various stages of the organism's development, or in different tissues, or as subjected to different environmental stresses. Although many of the details of these Experiments will be hidden from other subsystems, e.g., the specific technology used (chip vs. micro array vs. Northern blot), it is important to group the Profiles **70** generated by specific experiments, since comparing results across experiments can be very difficult in this area. As shown in

figures 8 and 9, the Expression subsystem will tie to the Metabolic Pathways subsystem via Proteins and Metabolites, and will tie to the Sequence subsystem via Genes or Transcripts.

[0086] In an embodiment of the present invention illustrated in figure 2, each bioinformatics component interacts directly only with the IS platform, which serves as the medium for the exchange of events and services, and permits components to remain decoupled. The exchange of events allows components to synchronize their behavior, and the exchange of services allows them to draw on one another's capabilities to retrieve data, perform analyses, or present user interfaces.

[0087] While the present invention has been described in the context of the preferred embodiment thereof, it will be readily apparent to those skilled in the art that other modifications and variations can be made therein without departing from the spirit or scope of the present invention. For example, the bioinformatic components may further comprise taxonomic information, bibliographic information, protein structure, signaling pathways, gene expression information etc. and still be considered claimed as part of the invention. Accordingly, it is not intended that the present invention be limited to the specifics of the foregoing description of the preferred embodiment, but rather as being limited only by the scope of the invention as defined in the claims appended hereto.

WHAT IS CLAIMED IS:

1. A computerized system for integrating object based data models comprising:

a software platform, one or more interface-based data models and one or more component services;

wherein the software platform comprises a Client Environment and a Client Bus;

wherein the software platform comprises software instructions in an object oriented programming language;

wherein the Client Environment comprises a common user interface;

wherein the Client Bus brokers all communication between software components and

further comprise an Event Channel function and a Services Broker function;

wherein the Event Channel enables components to register as listeners for particular types of events and react to them in prescribed ways;

wherein the Services Broker allows components to request and provide services to one another as defined by a method selected from the set of methods consisting of requesting service by name, requesting services by types, and requesting services by attributes for which services are requested;

wherein the components interoperate without having a direct knowledge of one another; and,

wherein the data models comprise at least one data model useful for bioinformatics research.

2. The system of claim 1 wherein the object oriented programming language further comprises CORBA Common Object Request Broker Architecture.

3. The system of claim 1 wherein the Services Broker maintains synchronization between Client Objects.

4. The system of claim 1 wherein the Client Bus comprise support for Dynamic Discovery.

5. The system of claim 1 wherein the Services Broker runs on the client computing platform.

6. The system of claim 1 wherein objects comprise a set of interface attributes belonging to each object that are dynamically established upon instantiation of the object.

7. The system of claim 1 wherein the data model comprises a high-level data map for an Integrated Bioinformation system.

8. The system of claim 1 wherein at least one component comprises a component outside the boundary of client space and wherein the outside component interfaces to the Client Bus via a Server Proxy.

9. The system of claim 8 wherein the outside component further comprises a Java Database Connectivity interface (JDBC).

10. The system of claim 8 wherein the outside component further comprises an object oriented programming interface.

11. A computerized system for integrating object based data models comprising:

a software platform, one or more interface-based data models and one or more component services;

wherein the software platform comprises a Client Environment and a Client Bus;

wherein the software platform comprises software instructions in an object oriented programming language;

wherein the Client Environment comprises a common user interface;

wherein the Client Bus brokers all communication between software components and

further comprise an Event Channel function and a Services Broker function;

wherein the Event Channel enables components to register as listeners for particular types of events and react to them in prescribed ways;

wherein the Services Broker allows components to request and provide services to one another as defined by a method selected from the set of methods consisting of requesting service by name, requesting services by types, and requesting services by attributes for which services are requested;

wherein the components interoperate without having a direct knowledge of one another; and,

wherein the data models comprise at least one high-level map for an Integrated Bioinformation system.

12. The system of claim 11 wherein the object oriented programming language further comprises CORBA Common Object Request Broker Architecture.

13. The system of claim 11 wherein the Services Broker maintains synchronization between Client Objects.

14. The system of claim 11 wherein the Client Bus comprise support for Dynamic Discovery.

15. The system of claim 11 wherein the Services Broker runs on the client computing platform.

16. The system of claim 11 wherein objects comprise a set of interface attributes belonging to each object that are dynamically established upon instantiation of the object.

17. The system of claim 11 wherein the data model comprises a high-level data map for an Integrated Bioinformation system.

18. The system of claim 11 wherein at least one component comprises a component outside the boundary of client space and wherein the outside component interfaces to the Client Bus via a Server Proxy.

19. The system of claim 18 wherein the outside component further comprises a Java Database Connectivity interface (JDBC).

20. The system of claim 18 wherein the outside component further comprises an object oriented programming interface.

21. A computerized system for integrating object based data models comprising:

a software platform, one or more interface-based data models and one or more component services;

wherein the software platform comprises a Client Environment and a Client Bus;

wherein the software platform comprises software instructions in an object oriented programming language;

wherein the Client Environment comprises a common user interface;

wherein the Client Bus brokers all communication between software components and

further comprise an Event Channel function and a Services Broker function;

wherein the Event Channel enables components to register as listeners for particular types of events and react to them in prescribed ways;

wherein the Services Broker allows components to request and provide services to one another as defined by a method selected from the set of methods consisting of requesting service by name, requesting services by types, and requesting services by attributes for which services are requested;

wherein the components interoperate without having a direct knowledge of one another; and,

wherein the data models comprise at least one high-level map for an Integrated Plant Bioinformation system.

22. The system of claim 21 wherein the object oriented programming language further comprises CORBA Common Object Request Broker Architecture.

23. The system of claim 21 wherein the Services Broker maintains synchronization between Client Objects.

24. The system of claim 21 wherein the Client Bus comprise support for Dynamic Discovery.

25. The system of claim 21 wherein the Services Broker runs on the client computing platform.

26. The system of claim 21 wherein objects comprise a set of interface attributes belonging to each object that are dynamically established upon instantiation of the object.

27. The system of claim 21 wherein the data model comprises a high-level data map for an Integrated Bioinformation system.

28. The system of claim 21 wherein at least one component comprises a component outside the boundary of client space and wherein the outside component interfaces to the Client Bus via a Server Proxy.

29. The system of claim 28 wherein the outside component further comprises a Java Database Connectivity interface (JDBC).

30. The system of claim 28 wherein the outside component further comprises an object oriented programming interface.

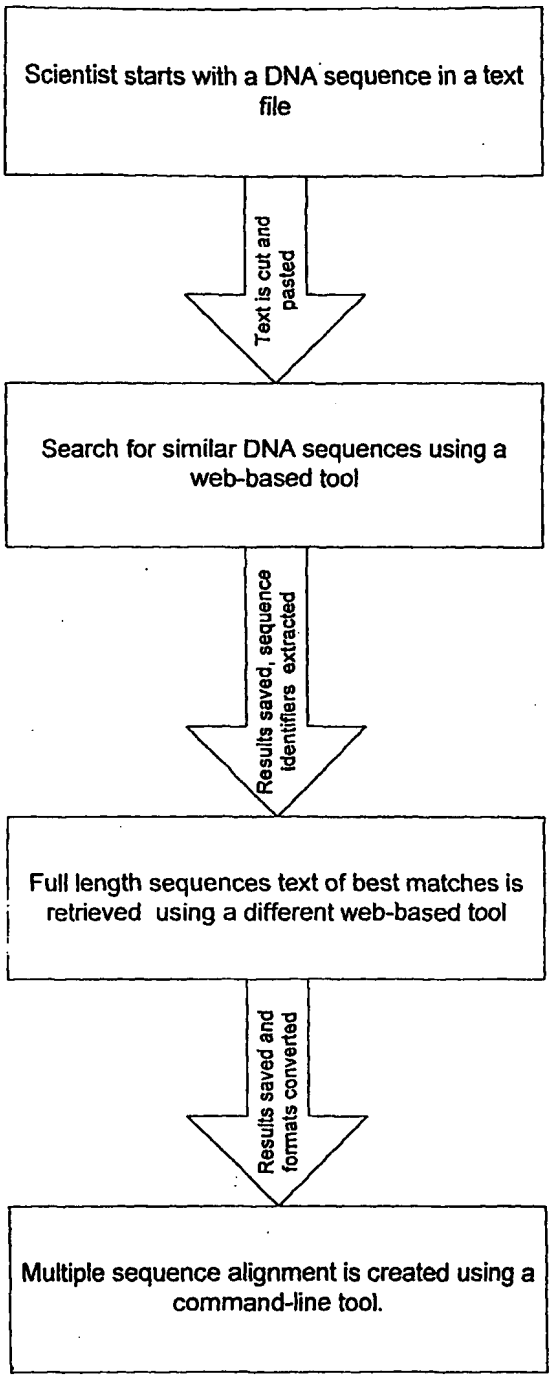

Scientist starts with a DNA sequence in a text file
Text is cut and pasted
Search for similar DNA sequences using a web-based tool
Results saved, sequence identifiers extracted
Full length sequences text of best matches is retrieved using a different web-based tool
Results saved and formats converted
Multiple sequence alignment is created using a command-line tool.

Figure 1

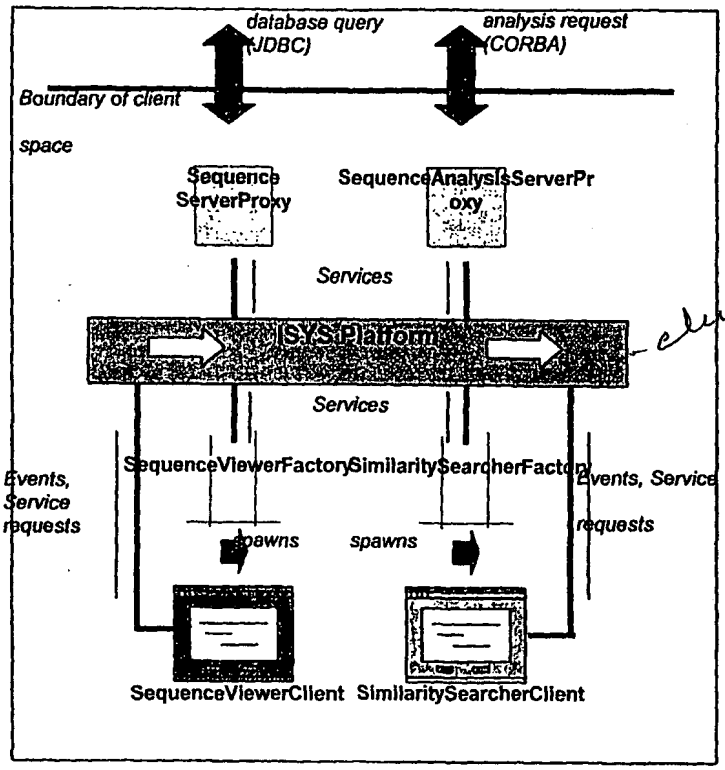

database query
(JDBC)
analysis request
(CORBA)
Boundary of client
space
SequenceServerProxy
SequenceAnalysisServerProxy
Services
S.Y.S. Platform
client Bus
Services
SequenceViewerFactory
SimilaritySearcherFactory
Events, Service requests
Events, Service requests
spawns
spawns
SequenceViewerClient
SimilaritySearcherClient

Figure 2

```
/**     Interface for "fundamental class" of DNA sequences. */
public interface IsysSequence {
        public String getICAccession();
        public String getSequenceText();
        public String getDescription();
}

/**     Interface for "fundamental class" of taxonomic nodes. */
public interface IsysTaxon {
        public String getRank();        // e.g., genus, species
        public String getName();        // e.g., Arabidopsis, thaliana
}

/**     Interface showing association with IsysTaxon. */
public interface HasTaxon { public IsysTaxon getTaxon(); }

/**     Simplified version of Homolog class used by SimilaritySearcher,
 *      which represents an apparent homolog to a "query sequence" based
 *      on the results of a similarity search.
*/
public class Homolog implements IsysSequence, HasTaxon {
        IsysSequence querySequence, backgroundSequence;
        Vector pairwiseAlignments;      // set of local alignments found by similarity
                                        // searching algorithm (e.g., HSPs in BLAST)
        // ...

        // methods specific to Homolog
        public getQuerySequence() { return querySequence; }
        // ...

        // IsysSequence methods.  From the point of view of ISYS, a
        // Homolog is representative of the background sequence.
        public String getICAccession() { return backgroundSequence.getICAccession(); }
        public String getSequenceText() { return backgroundSequence.getSequenceText(); }

        // HasTaxon method.  Similarly, implemented via background sequence.
        public IsysTaxon getTaxon() { return backgroundSequence.getTaxon(); }
}

/**     Example of event listener that processes visibility events for
 *      IsysSequences and IsysTaxons.  A listening component would
 *      register an instance of this class with the EventChannel (this
 *      listener is effectively an interface between components).
*/
class ItemVisibilityListener implements IsysEventListener {
        public void handleEvent(IsysEvent e) {
                Object o = e.getObject(), IsysTaxon t = null, IsysSequence s = null;

                if (o instanceof IsysTaxon)
                        t = (IsysTaxon)o;
                else if (o instanceof HasTaxon)
                        t = ((HasTaxon)o).getTaxon();

                if (o instanceof IsysSequence)
                        s = (IsysSequence)o;
                else if (o instanceof HasSequence)
                        s = ((HasSequence)o).getSequence();

                if (t != null) {
                        // Respond to event in terms of IsysTaxon
                }

                if (s != null) {
                        // Respond to event in terms of IsysSequence
                }
        }
}
```

Figure 3

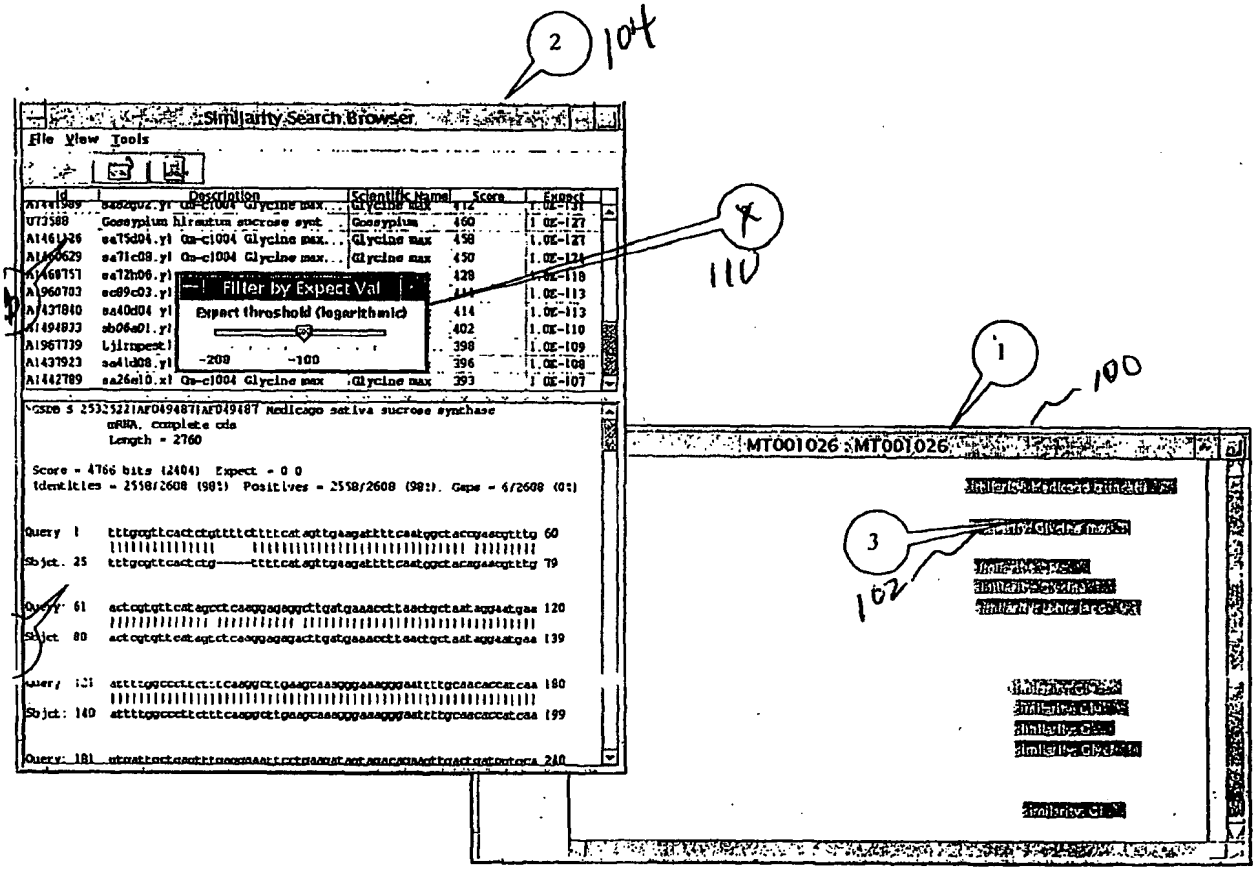

2
104
Similarity Search Browser
File View Tools
Description
Scientific Name
Score
Expect
Filter By Expect Val
Expect threshold (logarithmic)
-200
-100
110
1
100
MT001026 : MT001026
3
102

Figure 10 4

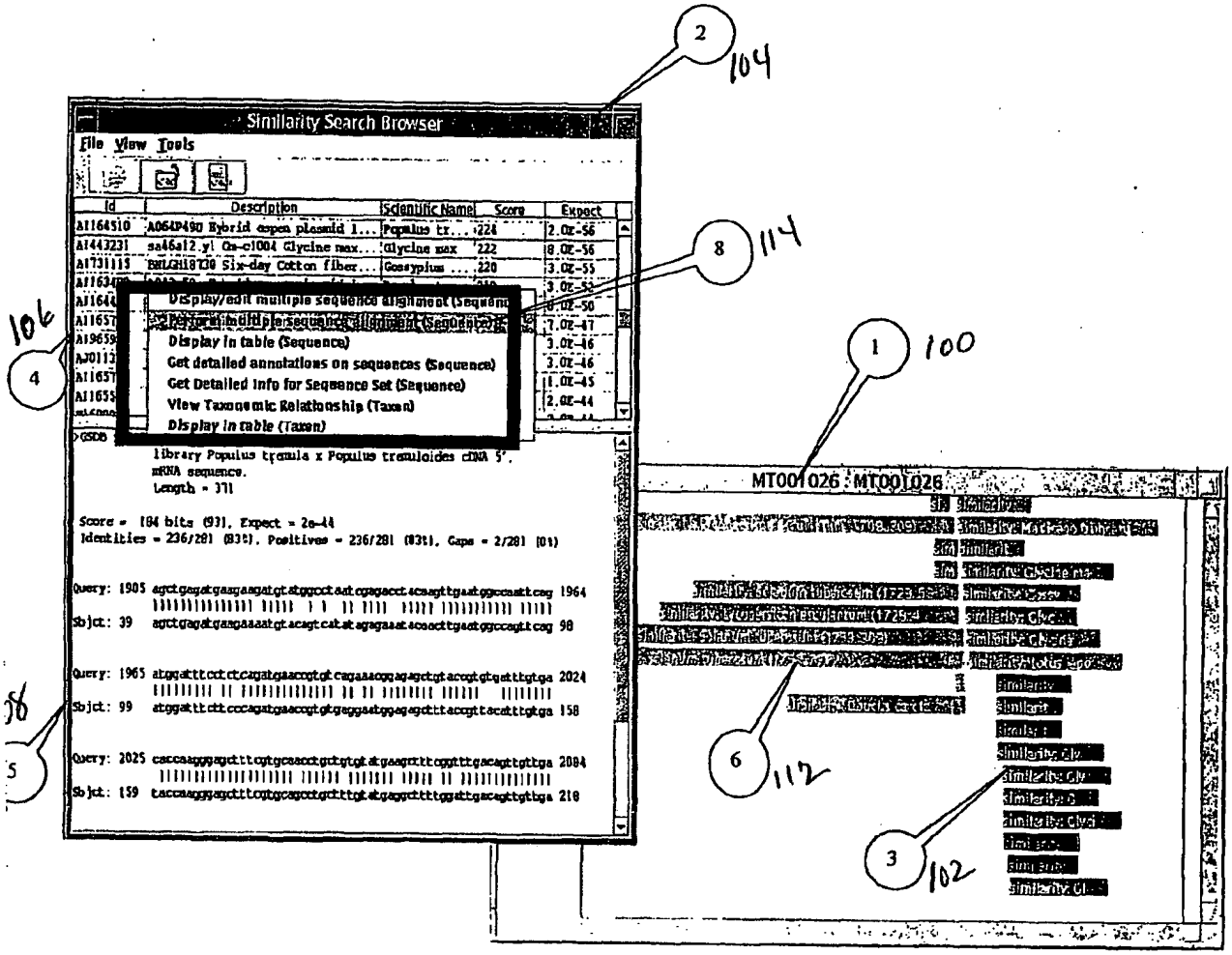

Similarity Search Browser
File View Tools
Id
Description
Scientific Name
Score
Expect
Display/edit multiple sequence alignment (Sequence)
Display in table (Sequence)
Get detailed annotations on sequences (Sequence)
Get Detailed Info for Sequence Set (Sequence)
View Taxonomic Relationship (Taxon)
Display in table (Taxon)
MT001026 : MT001026
2
104
8
114
106
4
1
100
6
112
3
102
5

Figure 1A 5

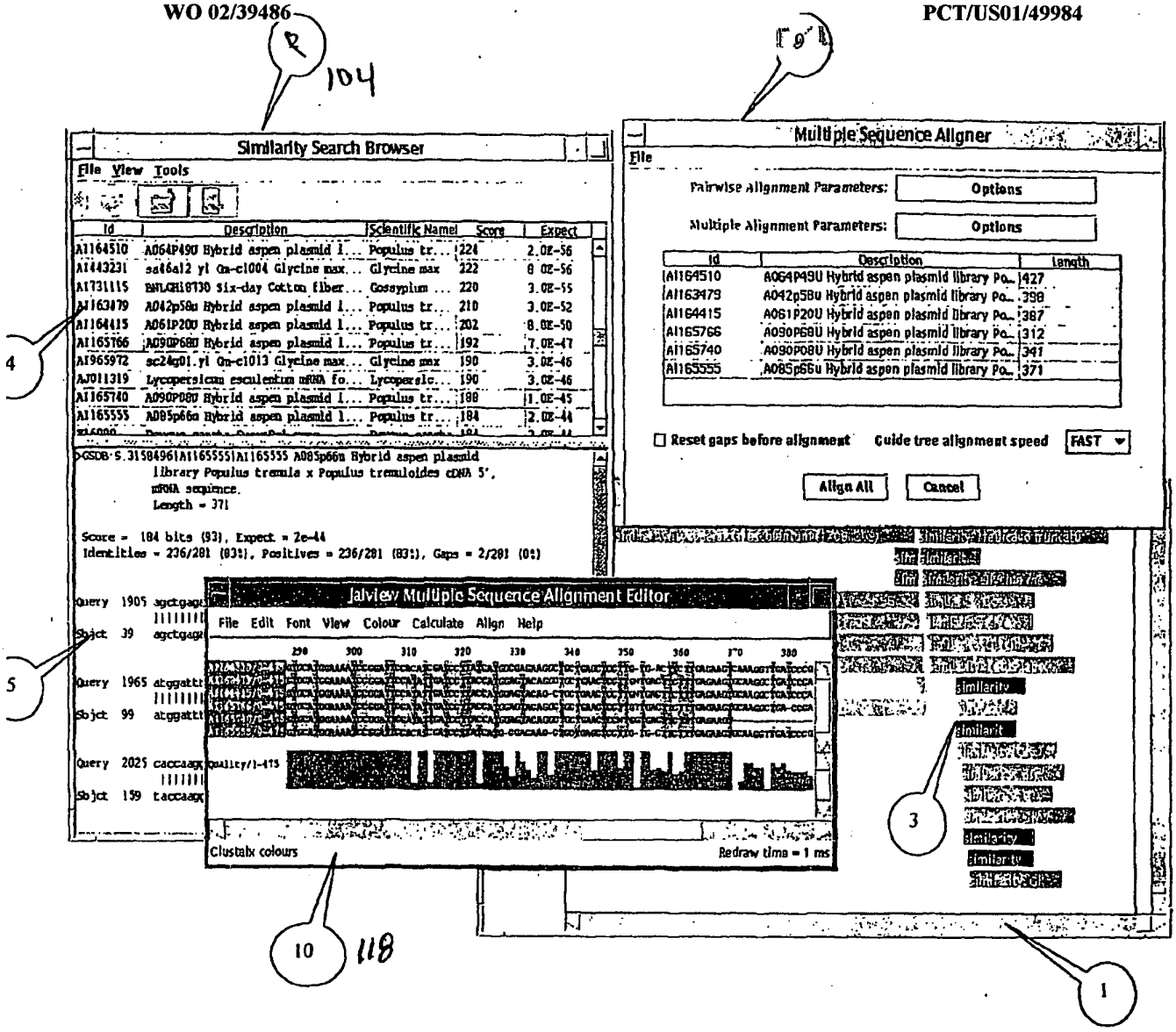
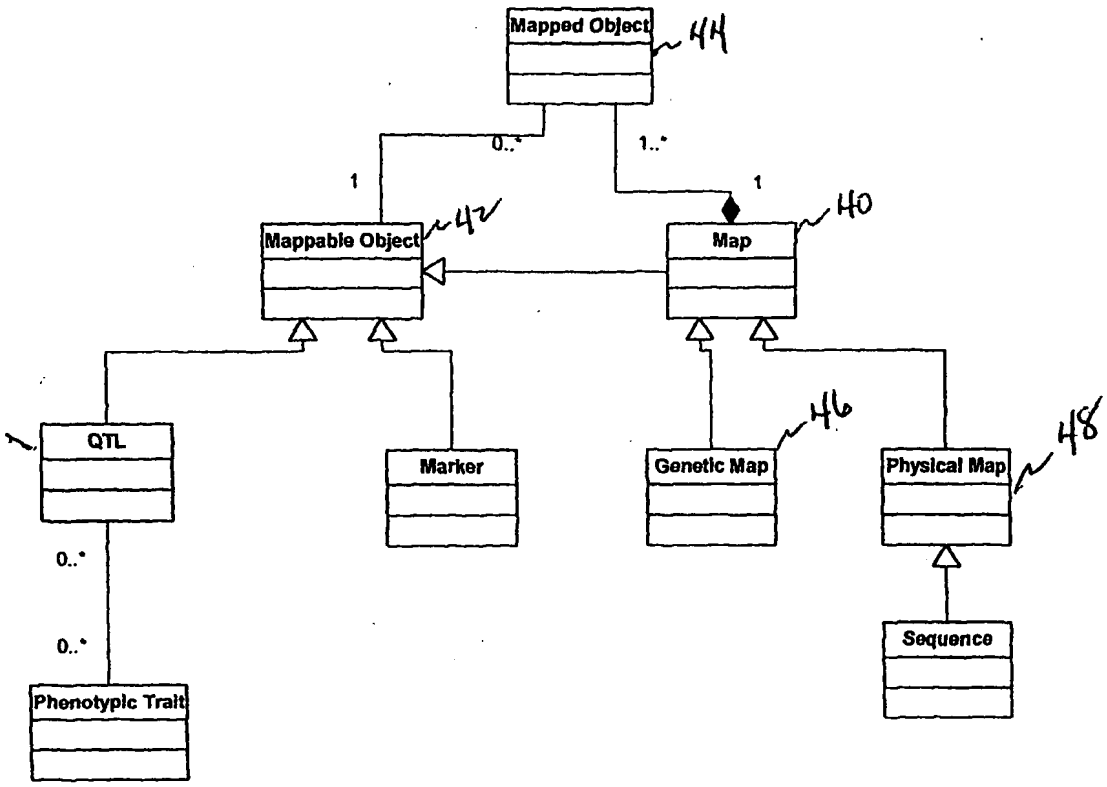
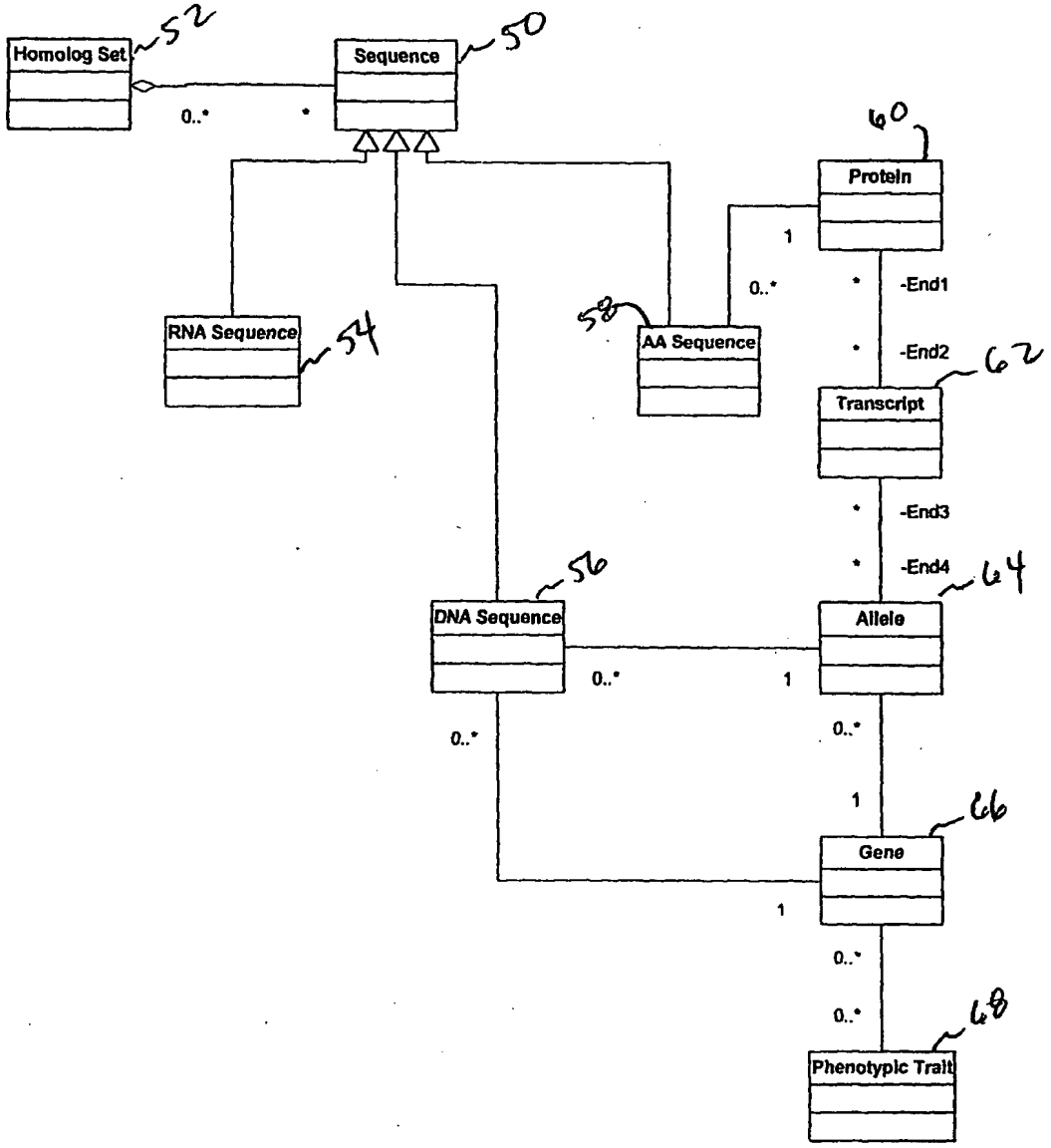

104
Similarity Search Browser
File View Tools
Id Description Scientific Name Score Expect
AI164510 A064P49U Hybrid aspen plasmid l... Populus tr... 224 2.0E-56
AI443231 sa46a12 yl Gm-c1004 Glycine max... Glycine max 222 8.0E-56
AI731115 BNLGHi8730 Six-day Cotton fiber... Gossypium ... 220 3.0E-55
AI163479 A042p58u Hybrid aspen plasmid l... Populus tr... 210 3.0E-52
AI164415 A061P20U Hybrid aspen plasmid l... Populus tr... 202 8.0E-50
AI165766 A090P68U Hybrid aspen plasmid l... Populus tr... 192 7.0E-47
AI965972 sc24g01.yl Gm-c1013 Glycine max... Glycine max 190 3.0E-46
AJ011319 Lycopersicon esculentum mRNA fo... Lycopersic... 190 3.0E-46
AI165740 A090P08U Hybrid aspen plasmid l... Populus tr... 188 1.0E-45
AI165555 A085p66u Hybrid aspen plasmid l... Populus tr... 184 2.0E-44
>GSDB:S:31584961|AI165555|AI165555 A085p66u Hybrid aspen plasmid
library Populus tremula x Populus tremuloides cDNA 5',
mRNA sequence.
Length = 371
Score = 184 bits (93), Expect = 2e-44
Identities = 236/281 (83%), Positives = 236/281 (83%), Gaps = 2/281 (0%)
Query 1905 agctgagc
Sbjct 39 agctgagc
Query 1965 atggattt
Sbjct 99 atggattt
Query 2025 caccaagg
Sbjct 159 taccaagg
Jalview Multiple Sequence Alignment Editor
File Edit Font View Colour Calculate Align Help
Clustalx colours
Redraw time = 1 ms
Multiple Sequence Aligner
File
Pairwise Alignment Parameters: Options
Multiple Alignment Parameters: Options
Id Description Length
AI164510 A064P49U Hybrid aspen plasmid library Po... 427
AI163479 A042p58u Hybrid aspen plasmid library Po... 398
AI164415 A061P20U Hybrid aspen plasmid library Po... 387
AI165766 A090P68U Hybrid aspen plasmid library Po... 312
AI165740 A090P08U Hybrid aspen plasmid library Po... 341
AI165555 A085p66u Hybrid aspen plasmid library Po... 371
Reset gaps before alignment
Guide tree alignment speed FAST
Align All
Cancel
4
5
3
10
118
1

Figure 12 6



7

Figure 4 - High-Level data model for Map Subsystem



8

Figure 8 High-level Data Model for Sequence Subsystem

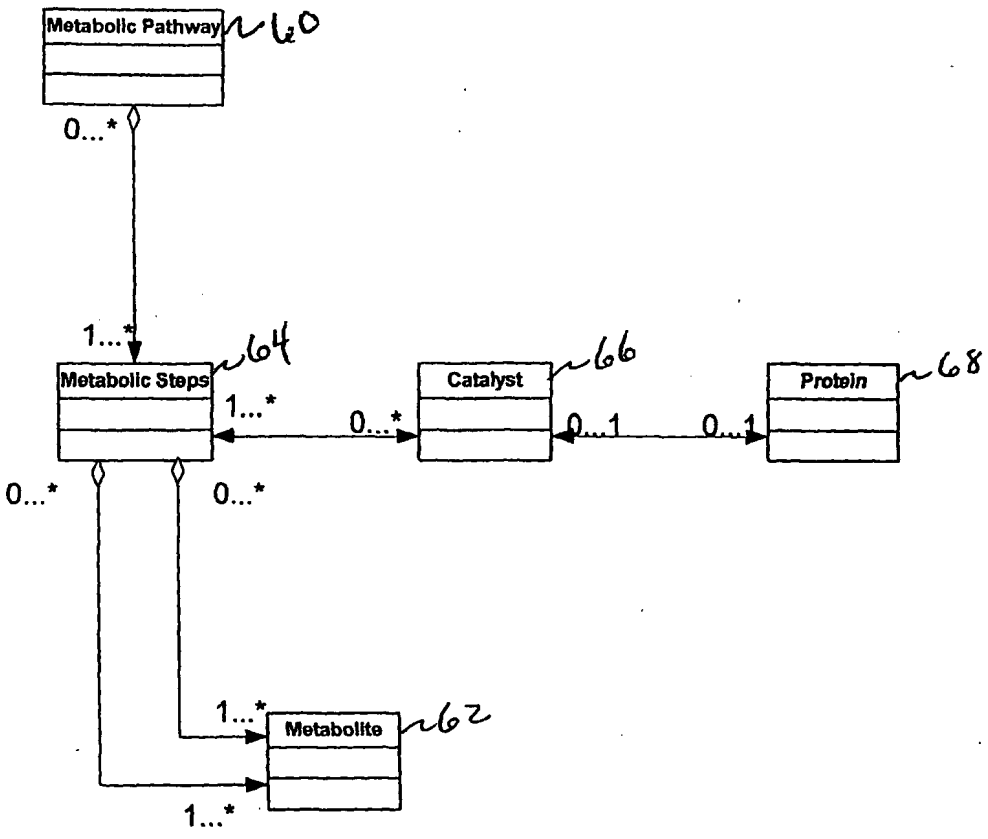
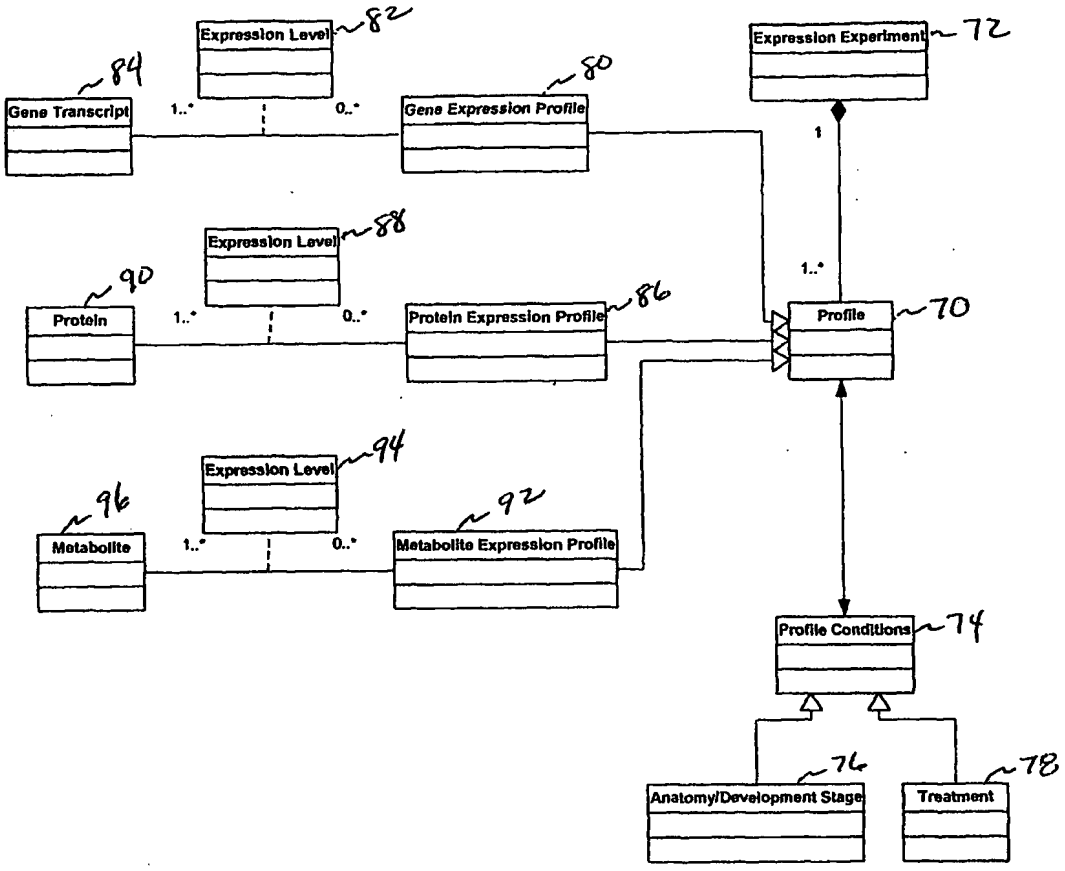


Figure 9 High-level Data Model for Metabolic Pathway Subsystem



10

Figure 1 High-level Data Model for Gene Expression Subsystem

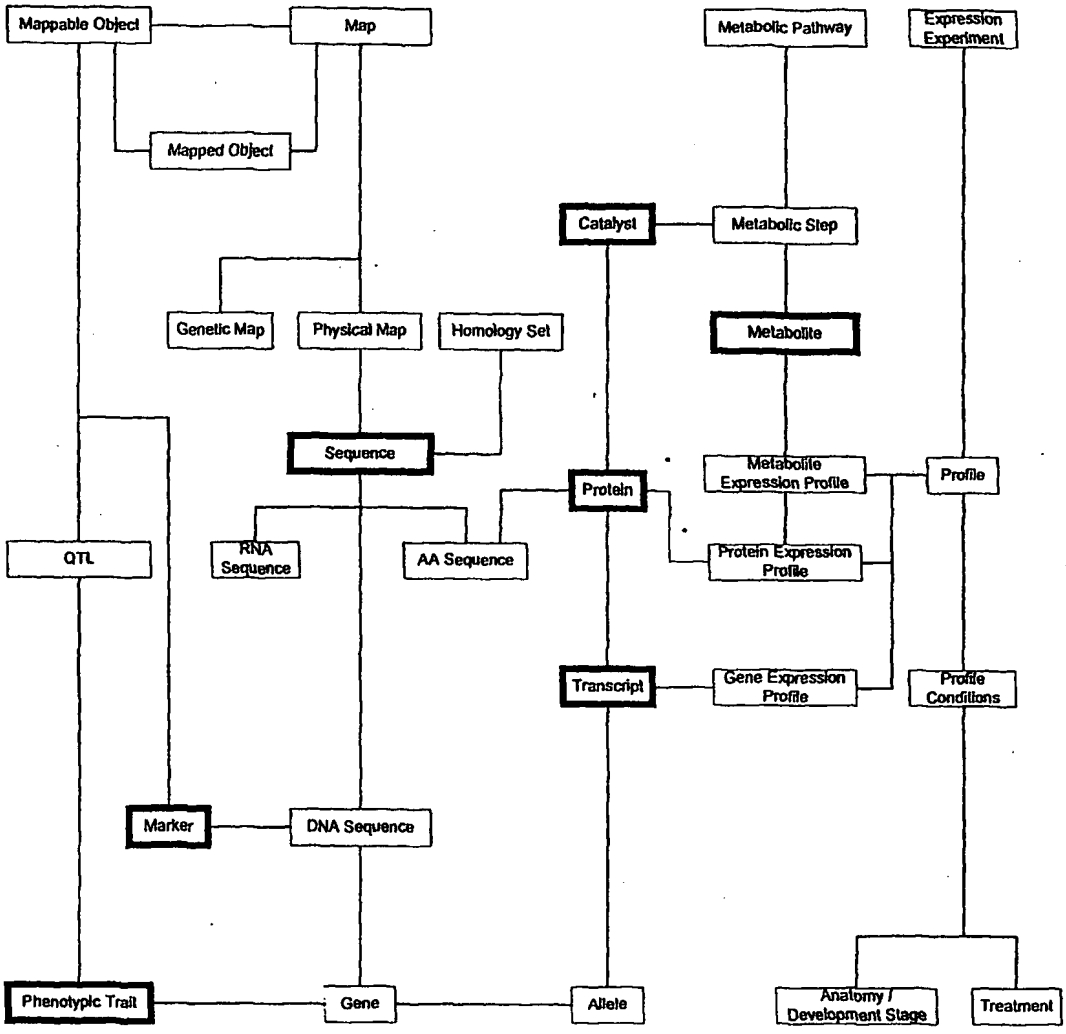


Figure 8 Integrated Data Model including Map, Sequence, Pathway and Expression Subsystems